

**This Page Is Inserted by IFW Operations
and is not a part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- **BLACK BORDERS**
- **TEXT CUT OFF AT TOP, BOTTOM OR SIDES**
- **FADED TEXT**
- **ILLEGIBLE TEXT**
- **SKEWED/SLANTED IMAGES**
- **COLORLED PHOTOS**
- **BLACK OR VERY BLACK AND WHITE DARK PHOTOS**
- **GRAY SCALE DOCUMENTS**

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

AL

⑨ 日本国特許庁(JP)

⑩ 特許出願公開

⑫ 公開特許公報(A)

昭64-72237

⑤ Int.Cl.⁴

G 06 F 9/32
9/26

識別記号

3 2 0
3 2 0

庁内整理番号

D-7361-5B
A-7361-5B

⑬ 公開 昭和64年(1989)3月17日

審査請求 未請求 発明の数 1 (全7頁)

⑭ 発明の名称 アドレス計算方式

⑮ 特 願 昭62-228337

⑯ 出 願 昭62(1987)9月14日

⑰ 発 明 者 海 永 正 博 神奈川県川崎市麻生区王禅寺1099番地 株式会社日立製作所システム開発研究所内
⑰ 発 明 者 橋 本 幸 治 東京都小平市上水本町1479番地 日立マイクロコンピュータエンジニアリング株式会社内
⑱ 出 願 人 株式会社日立製作所 東京都千代田区神田駿河台4丁目6番地
⑱ 出 願 人 日立マイクロコンピュータエンジニアリング株式会社 東京都小平市上水本町1479番地
⑲ 出 願 人 坂 村 健 東京都港区白金台3-12-30-105
⑳ 代 理 人 弁理士 小川 勝男 外1名
最終頁に続く

明 細 書

1. 発明の名称

アドレス計算方式

2. 特許請求の範囲

(1) 第1類アドレス指定フィールドで指定された値を計算途中アドレスの初期値とし、連続的に配置された複数の第2類アドレス指定フィールドの指示に従い順次に計算途中アドレス計算し、最終アドレスを求めることを特徴とするアドレス計算方式。

(2) 第1項記載のアドレス計算方式において、第2類アドレス指定フィールドの指定によりアドレス加算の1つのオペランドとしてプログラムカウンタの内容を指定できることを特徴とするアドレス計算方式

(3) 第1項記載のアドレス計算方式において、第2類アドレス指定フィールドの指定により計算したアドレスのロケーション内容を新たな計算途中アドレスとする場合に、該ロケーションのサイズを情報処理装置の語サイズの半分また

は4分の1に指定できることを特徴とするアドレス計算方式。

3. 発明の詳細な説明

〔産業上の利用分野〕

本発明は情報処理装置において、機械命令オペランドのアドレス計算方式にかかわる。

〔従来の技術〕

情報処理装置の機械命令として、無条件分岐命令(JMP命令)というものがある。これはオペランドで指定されたアドレスへの無条件分岐を引き起こす命令である。

無条件分岐命令では、1つのアドレスにしか分岐できないが、多方向分岐命令を備えた情報処理装置も知られている。たとえば、共立出版社発行のVAX-11 アーキテクチャ ハンドブックによれば、VAX-11は多方向分岐命令(CASE命令)を提供している。この命令機能は以下の通りである。

まず使用例は以下になる。ここで、dispnは各々2バイトサイズである。

```

CASE selector,base,limit
disp[0]
...
disp[limit]

```

そして、この命令発行時の動作は第2図のマイクロプログラムにより定義される。第2図において個別の動作は、プログラミング言語Cに準じて、いわゆる高級言語の水準で記述してあるが、本来はハードウェアの個別ゲートを制御するいわゆる低水準の処理に分解されよう。しかし、ここでは、いたずらに処理内容を詳細化し、それにより本発明の本質を理解しやすくすることを避けるために、また個別ゲートの制御自体は本発明の範囲外であるので、第2図で記述された水準の処理ステップで、この動作を説明することにする。

第2図において、Tはある作業レジスタ、selector,base,limitは命令オペランドを受ける作業レジスタ、PCはプログラムカウンタである。また'<='は比較(小、または一致)演算子、disp

5において、'exit()'によりマイクロプログラム処理の終了を命令デコーダに通知することにより、CASE命令処理が終了する。

尚、ステップ203を経由することは、dispの第T(selector-base)番目の内容にもとずいて多方向分岐することに他ならない。一方ステップ204を経由することは、T(selector-base)の値が望まれた範囲外であって、従ってdisp配列の直後のアドレスに抜けることに他ならない。

以上のようにして、CASE命令を用いて多方向分岐を実現できる。

[発明が解決しようとする問題点]

以上示したように、CASE命令を用いると多方向に分岐できてなるほど便利である。しかし、この命令では分岐しかできない。多方向分岐の他に、多方向メモリアクセスというものも考えられる筈であるが、CASE命令では多方向分岐しかできない訳である。

この対策として、個別の命令、例えばメモリ内容反転命令(NEG命令としておく)に対応して、

[T]は第T番目のdisp要素の参照、'<<'は左シフト演算子をそれぞれ表現する。尚、VAX-11のCASE命令実行時のプログラムカウンタPCの値は最初のdisp要素のアドレスを指している。

命令デコーダが命令ストリーム中にCASE命令を検出すると、命令デコーダは第2図に示す処理フローのマイクロプログラムを起動する。まずステップ201において、selectorの値からbaseの値を減算し、結果の値を一時レジスタTにセットする。次にステップ202において、一時レジスタTの値とlimitの値を比較する。そして、Tの値がlimitの値以下であれば、次にステップ203において、PCの値とdispの第T番目の内容を加算し、その結果の値でPCを更新し、そしてステップ205にジャンプする。一方、Tの値がlimitの値以下でなければ、ステップ204にジャンプし、ステップ204において、PCの値とlimitの値を1ビット左シフトした値を加算し、その結果の値でPCを更新し、ステップ205に進む。そして、ステップ20

多方向反転命令(CASE_NEG命令としておく)を用意するという対策もありえるが、この対策は命令の数が増えず、現実的な対策とはいえない。

本発明の目的は、JMP命令で多方向分岐を実現できるといった柔軟なアドレス計算方式を実現することにある。

[問題点を解決するための手段]

そこで、本発明では、CASE命令のアドレス計算部分をCASE命令の機能から分離させ、そしてオペランドのアドレス計算の機能に含ませることにした。

[作用]

これにより、CASE命令は不要となった。というのは、JMP命令の分岐アドレス指定において、CASE命令の機能から分離されたCASE命令のアドレス計算部分を指定するようにすれば良いからである。

勿論、本発明により、CASE_NEG命令の機能はNEG命令で実現できる。

[実施例]

以下、本発明の1実施例を詳細に説明する。

第5図は本発明を実施した情報処理装置の構成図である。主メモリ510には機械語プログラム511が格納されており、またデータ領域512が配置される。CPU520は主メモリにアクセスしつつ機械語プログラム511の指示に従いアクセスすべきアドレスを計算しデータ領域512の値を参照、更新することによりデータ処理を遂行する。さらに機械語プログラム511の指示によっては、入出力装置530を介して情報処理装置と外界とのやりとりが行われる。読みだし専用メモリ521内にはマイクロプログラムが格納される。このマイクロプログラムは、機械語の指示などを解釈、実行するためのプログラムである。

第3図はオペランドを1つ持つ命令の本実施例による形式を表現している。フィールド301は命令コードが格納される部分である。フィールド302はオペランドの指定法が格納される部分である(第1類アドレス指定フィールド)。

```

1: index= PC
i : 間接指定子
    0: 間接参照なし
        tmp= tmp+(index<<xx)+disp
    1: 間接参照あり
        tmp= *(tmp+(index<<xx)+disp)
s : サイズ指定子
    0: 間接参照時、4nビット操作
    1: 間接参照時、2nビット操作
e : 継続指定子
    0: アドレス計算終了
    1: 次の追加フィールドのアドレス
        計算に進入
  
```

さて、この追加フィールドを具体的に解釈実行するマイクロプログラムを第1図に示す。この図で、curpは現追加フィールド指し示す作業レジスタとし、命令デコードが正しく値をセットしているものとする。PCは現在実行中の命令アドレスを指すいわゆるプログラムカウンタである。GRnはcurpが指す追加フィールド内の部

オペランドの指定法によっては、後続の位置に追加フィールド(第2類アドレス指定フィールド)が配置される。以下、本発明の説明に必要な範囲で、オペランド指定法を示しておく。

```

フィールド2 意味
0001 n 第n汎用レジスタを
        オペランドとする
0000 1111 PCの値を途中アドレス(tmp)
        の初期値とし、追加フィールド
        の指定に従いアドレス計算
  
```

追加フィールドの構造は第4図の通りであり、個別のフィールドの意味は以下の通りである。尚、計算途中アドレスをtmpで表現する。また、indexという変数を導入する。

```

disp: オフセット(iフィールド参照)
xx : シフトカウンタ(iフィールド参照)
n : 汎用レジスタ番号(iフィールド参照)
m : インデックス指定子
    0: nフィールドで指定した汎用レジスタ
    index= GRn
  
```

分フィールドnで特定される汎用レジスタである。'->'はポインタオフセット演算子であり、例えば'curp->m'でcurpが指す追加フィールド内m部分フィールドの参照を表現する。'?',':'の2つで条件評価演算子となり、例えば'(curp->m==1)?PC:GRn'で、curp->mの値が1か検査し、そうであればPCの内容を式の値とし、そうでなければGRnの内容を式の値とすることを表現する。'*(short*)'、'*(int*)'で間接参照を表現している。例えば'*(short*)tmp'でtempの値が指している2バイトメモリロケーションへの参照を表現し、同様に'*(int*)tmp'でtempの値が指している4バイトメモリロケーションへの参照を表現している。また、'++'は+1演算子であり、例えば'curp++'でcurpが次の追加フィールドを指すことになる。

第2図のマイクロプログラムソーステキストと同じく、この第1図のマイクロプログラムソーステキストをコンパイルまたはアセンブルして0/1のビットパターンで構成される2進のマイ

クロプログラムとし、これを第5図のマイクロプログラム格納用の読みだし専用メモリ521に格納すれば、本発明が情報処理装置に実際に組み込まれることになるのは勿論である。尚、第1図のフローチャート表現が第6図となる。

さて、本発明によって、JMP命令で多方向分岐が効率的に行えることを示すためにJMP命令とそれに関連した情報が以下の形でメモリロケーションに配置されているものとする。そして、汎用レジスタGR0の値に応じて、L0, L1, ..., L10のアドレスに多方向分岐するものとする。尚GR0の値の範囲は0~10が保証されているものとする。

Lb:	JMP	00001111
	第1追加フィールド	
	第2追加フィールド	
	L0-Lb	
	L1-Lb	
	L10-Lb	

フィールドのnサブフィールドの値が0であるので、GR0の値を作業レジスタindexにセットする。次にステップ103で、tmpの値と、indexの値を現追加フィールドのxxサブフィールドの値(実は1)で左シフトした値と、現追加フィールドのdispサブフィールドの値(実はL0-Lb)の3者を加算し、その結果の値をtmpに、セットする。このとき、tmpの元々の値はLbであり、またdispはL0-Lbであり、従ってtmp+dispは実はL0である。従って、ステップ103の結果tmpにセットされた値は、じつは第2追加フィールド以降に配置されたオフセット配列のGR0の値番目のアドレスを指している。次にステップ104で、現追加フィールドのiサブフィールドの値が1であるか検査し、今の場合1でないので、ステップ105に抜ける。ステップ105では、現追加フィールドのsサブフィールドの値が1であるか検査し、今の場合1であるので、tmpが指す2バイトロケーションの内容を読みだし、それをtmpにセットする。次にステップ106

また、2つの追加フィールドの詳細は以下の通りとする。

ei	n	msxx	disp
11	0	00	1 L0-Lb
00	0	10	0

第1追加フィールド

第2追加フィールド

この2つの追加フィールドが定めたtmp計算は以下の通りとなろう。

第1追加フィールド: tmp = (tmp + (GR0 << 1) + L0 - Lb)

第2追加フィールド: tmp = tmp + (PC << 0) + 0

さて、命令デコーダがアドレスLbのJMP命令を検出したものとする。このとき、命令デコーダはまず、オペランドのアドレス計算を行う必要がある。そして、今の場合オペランドフィールドのボタンが'00001111'であり、従って、作業レジスタtmpにPCの値をセットして、第1図のマイクロプログラムを起動する。ステップ101の検査は常に成功終了して、従って次にステップ102に進む。ステップ102では現追加フィールドのnサブフィールドの値が1か検査し、今の場合1ではなく、さらに現追加フィ

で、現追加フィールドのeサブフィールドの値が0であるか検査し、今の場合0でないので、ステップ107に抜ける。ステップ107で、次の追加フィールドをcurpが指すように更新し、そしてステップ101にジャンプすることにより、次のループ処理を開始する。

ステップ101の検査は常に成功終了して、従って次にステップ102に進む。ステップ102では現追加フィールドのnサブフィールドの値が1か検査し、今の場合1であるので、PCの値を作業レジスタindexにセットする。次にステップ103で、tmpの値と、indexの値を現追加フィールドのxxサブフィールドの値(実は0)で左シフトした値と、現追加フィールドのdispサブフィールドの値(実は0)の3者を加算し、その結果の値をtmpにセットする。このとき、tmpの元々の値は実は第2追加フィールド以降に配置されたオフセット配列のGR0の値番目のロケーション内容であり、従って、ステップ103の結果tmpにセットされた値は、実はL

0, L1, ..., L10の内のGR0で特定されるアドレスと
いうことになる。次にステップ104で、現追加
フィールドのiサブフィールドの値が1であ
るか検査し、今の 合1であるので、ステップ
106にジャンプする。次にステップ106で、
現追加フィールドのeサブフィールドの値が0
であるか検査し、今の場合0でないので、'exit
()'によりアドレス計算処理を終了し、その旨
命令デコーダに通知する。通知を受けた命令デ
コーダは、今度は命令処理を遂行する必要があ
る。本実施例における命令デコーダは分岐命令
の命令処理を自分自身で行うとする。この場合
tmpの値を命令デコーダ自身がプログラムカウ
ンタPCにセットすることにより、JMP命令の命
令処理が終了することになる。

さて、この新PCはL0, L1, ..., L10の内のGR0で特
定されるアドレスであり、従ってJMP命令によ
り多方向分岐命令の機能が実現できたことにな
る。

【発明の効果】

要素サイズを小さくすることにより、全体のメ
モリサイズを小さくできることになる。

4. 図面の簡単な説明

第1図は本発明の実施例によるアドレス計算
のマイクロプログラムフローを示す図、第2図
は従来技術によるCASE命令機能の実現例を示す
図、第3図は本発明の実施例で想定した1オペ
ランド命令のフィールド分割図、第4図は本発
明によるアドレス計算用追加フィールドのサブ
フィールド分割図、第5図は本実施例による情
報処理装置の構成図、第6図は第1図を所謂フ
ローチャートで表現した図である。

以上示したように本発明により、JMP命令で
多方向分岐命令の機能を実現できる。またJMP
命令の代わりに、NEG命令を用いれば勿論CASE_
NEG命令の機能を実現できることも明らかであ
る。

本実施例の説明において、第2の追加フィ
ールドの指定により、PCの値をアドレス計算に用
いた。このときのPCの値は、最終アドレスを求
めるためのベースアドレスとして用いられてい
るものと解釈できる。そして、第1の追加フィ
ールドの指定により求めたtmpの値は、ベース
アドレスからのオフセットとして解釈できる。
勿論、第2追加フィールドの後続の位置に配置
された配列の個別要素は、本実施例の説明にお
いては、まさしくL10-L0などのオフセットが配
置されていたのであったが、オフセットであ
れば4バイトサイズである必要はなく、例えば本
実施例のように2バイトサイズでもよく、ま
た1バイトサイズであっても構わない。そして、
オフセット配列の要素数が多い状況であれば、

第 1 図

```

while(1){                                ---101
    index=(curp->m==1)?PC:GRn;          ---102
    tmp=tmp+(index<<(curp->xx))+disp;    ---103
    if(curp->i==0) goto L;                ---104
    tmp=(curp->s==1)?                      ---105
        *(short*)tmp: *(int*)tmp;
    L:if(curp->e==0) exit();              ---106
    curp++;                               ---107
}

```

代理人弁護士 小川 勝 男

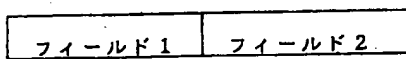
第 2 図

```

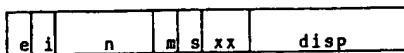
T = selector-base;      -----201
if( T <= limit)         -----202
    PC = PC + disp[T];  -----203
else
    PC = PC + (limit<<1); -----204
exit();                 -----205

```

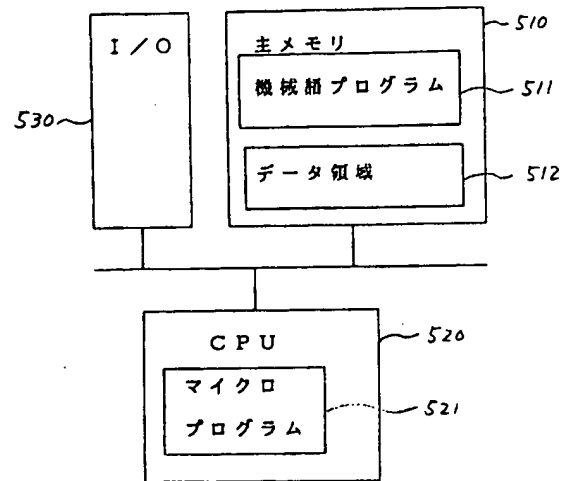
第 3 図



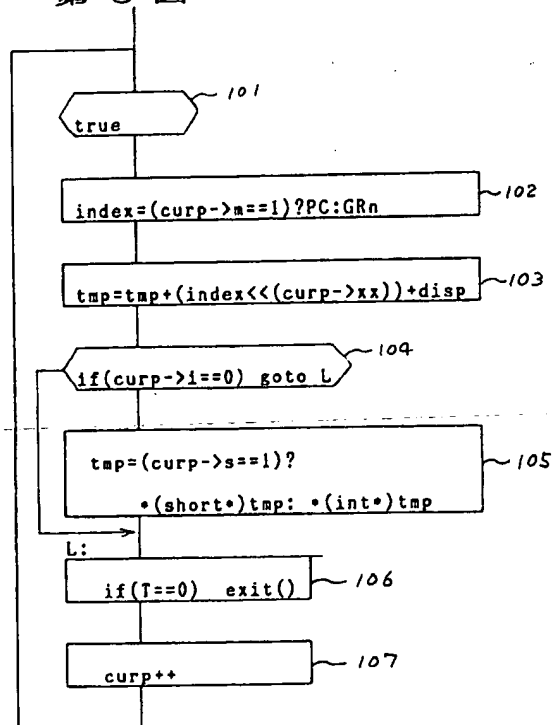
第 4 図



第 5 図



第 6 図



第1頁の続き

⑫発明者	川崎	郁也	東京都小平市上水本町1450番地 株式会社日立製作所武蔵工場内
⑫発明者	長谷川	淳	東京都小平市上水本町1479番地 日立マイクロコンピュータエンジニアリング株式会社内
⑫発明者	坂村	健	東京都港区白金台3-12-30-105
⑫発明者	秋田	英彦	神奈川県川崎市麻生区王禅寺1099番地 株式会社日立製作所システム開発研究所内